(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2002 (10.05.2002)

PCT

(10) International Publication Number
WO 02/37731 A2

(51) International Patent Classification[7]: H04L

(21) International Application Number: PCT/IL01/01017

(22) International Filing Date:
1 November 2001 (01.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/704,757 3 November 2000 (03.11.2000) US

(71) Applicant *(for all designated States except US)*: **CUTE LTD.** [IL/IL]; 9 Ha'Arad Street, 69 710 Tel Aviv (IL).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **GOLDBERGER, Jacob** [IL/IL]; 5 Elisha Street, Givataim 53 604 (IL).

(74) Agent: **G. E. EHRLICH (1995) LTD.**; 28 Bezalel Street, 52 521 Ramat Gan (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/37731 A2

(54) Title: DECODING OF LOW DENSITY PARITY CHECK CODES

(57) Abstract: A method of decoding a data signal using a low density parity check matrix, comprising a first stage of iteratively carrying out horizontal and vertical decoding steps to form a matrix $\{\delta q_{ik}\}$ and a second stage of testing for parity, characterized in that the elements of $\{\delta q_{ik}\}$ are updated sequentially, such that each element $\{\delta q_{ik}\}$ is employable for subsequent calculations as soon as it is updated.

DECODING OF LOW DENSITY PARITY CHECK CODES

Field of the Invention

The present invention relates to decoding of low density parity check codes and more particularly but not exclusively to a method and apparatus for decoding of low density parity check codes that is suitable for digital data communication such as multimedia.

Background of the Invention

Error correcting codes are widely utilized to obtain reliable communications over noisy channels. Generally speaking, a linear error correcting code C can be described by a parity-check matrix H satisfying $Hx = 0$ for any codeword $x \in C$. H is an $m \times n$ matrix where n is the size of a codeword and m is the number of linear constraints that must be satisfied by each code word. Each row of H therefore represents a linear homogeneous parity-check equation. Low-density parity-check (LDPC) codes are a particular class of linear error correcting codes characterized by a highly sparse parity-check matrix. Typically, in a matrix having a relatively large row length, the entire row may consist of just three ones, the remainder being zeroes.

LDPC codes were originally introduced and investigated by Gallager in 1962, c.f. R.G. Gallager , "Low-density parity-check codes", IRE Trans. Info. Theory, vol. IT-8, pp 21-28, 1962, the contents of which are hereby incorporated by reference.

A problem with the use of low density parity check matrices has been to provide a method of decoding, and one of the most significant features of Gallager's work is the introduction of iterative decoding algorithms. He showed that, when applied to sparse parity-check matrices, such algorithms are capable of achieving a significant fraction of the channel capacity at relatively low complexity. Furthermore, the number of computations per bit per iteration is independent of the block length n.

Since Gallager's prominent contribution, LDPC codes have been rediscovered and further investigated by Tanner, Wiberg, Mackay and Neal and others. Details of these investigations may be found in R.M. Tanner, "A recursive approach to low complexity code", IEEE transactions on information theory, 27)5), pp 533-547, 1981, and D.J.C MacKay and R.M. Neal, "Near Shannon limit performance of low-density parity-check codes", Electronic letters, vol. 32, pp. 1645-1646, 1996, the contents of which are hereby incorporated by reference.

Important modifications to Gallager's codes include:

1) Davey and MacKay, cited in M.C. Davey and D.J.C Mackay, "Low-density parity-check codes over GF(q)", IEEE communication letters, vol.2 , No 6 m, 1998, the contents of which are hereby incorporated by reference. In this citation, there is proposed a non-binary version of LDPC codes;

2) Gallager considered regular codes whose parity-check matrix had fixed row and column weights. These constraints can be relaxed to produce irregular LDPC codes having a variety of row and column weights as described

by Mitzenmacher *et. al.* cited in M. G. Luby, M.Mitzenmacher, M.A. Shokrollahi and D.A. Spielman, "Improved low density parity-check codes using irregular graphs and belief propagation", proceedings of the IEEE International Symposium on Information Theory (ISIT) , pp. 117, 1998, the contents of which are hereby incorporated by reference..

Known decoders include the maximum likelihood decoder and the soft output decoder. The more widely used of the two, the maximum-likelihood decoder involves finding a most probable codeword (where the likelihood of the codeword is dependent on the channel model). The soft-output decoder differs from the maximum likelihood decoding in that it provides an *a posteriori* probability for each symbol of the codeword. A problem, however, is that the soft-output decoder is typically more computationally involved than the maximum-likelihood decoder.

Gallager therefore proposed the iterative decoding scheme referred to above, based on the (later termed) *belief propagation method*, details of which are discussed in J. Pearl , "probabilistic reasoning in intelligent systems: Networks of plausible inference", Morgan Kaufmann, 1988, the contents of which are hereby incorporated by reference. In this citation the belief propagation method, which approximately converges to the *a posteriori* probability of each symbol, is explained. The method relies on a graph-based representation of codes, where the decoding can be understood as message passing in a factor graph. Belief propagation produces exact probabilities in case of a non-cyclic graph. Unfortunately, the graph associated with an LDPC

code is cyclic and therefore belief propagation may produce inaccurate probabilities. Nevertheless, Gallager's decoding algorithm gives good empirical performance since, in particular, the end product is the decoding, and so the posterior probabilities need not necessarily be exact.

A brief description of Gallager's iterative algorithm follows. For brevity of exposition, we consider the binary case. The extension to the non-binary case is straightforward.

Gallager's iterative decoding for LDPC codes

Gallager's algorithm has two alternating components commonly referred to as the horizontal and vertical steps. More specifically, two binary distributions, q_{ik} and r_{ik}, associated with the non-zero elements h_{ik} of the sparse parity-check matrix H, are iteratively updated. The quantity $q_{ik}(0)$ represents the probability that the k`th bit of the transmitted codeword is zero given the information obtained from all the parity equations other than the i`th equation. In a similar manner, $r_{ik}(0)$ represents the probability that the i`th parity-check equation is satisfied given that the k`th bit is zero and all the other bits are statistically independent, with associated distributions $q_{i1}\cdots q_{in}$. Assuming that the codewords are used with equal probability on an arbitrary binary-input continuous-output channel, Gallagers algorithm can be described as follows.

1. Initialization

Denote by $p_k(0)$ the prior probability that the k`th bit of the transmitted codeword is zero. This probability can be calculated from the received vector corresponding to the transmitted codeword and the channel model. For each non-zero entry h_{ik} of the parity-check matrix H, set:

$$q_{ik}(0) = p_k(0) \;\; ; \; q_{ik}(1) = p_k(1).$$

2. Horizontal step

For each non-zero entry h_{ik} of the matrix H:

define $\delta q_{ik} = q_{ik}(0) - q_{ik}(1)$ and compute

$$r_{ik}(0) = \frac{1}{2}(1 + \prod_{l \neq k} \delta q_{il}) \qquad (1)$$

$$r_{ik}(1) = \frac{1}{2}(1 - \prod_{l \neq k} \delta q_{il}) \qquad (2)$$

where l runs over the non-zero bit positions of the i`th parity equation (i.e. $h_{il} \neq 0$), excluding the k`th position.

3. Vertical Step

For each non-zero entry h_{ik} of the matrix H, update q_{ik} in the following manner:

$$q_{ik}(0) = \alpha \; p_k(0) \prod_{j \neq i} r_{jk}(0) \qquad (3)$$

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} r_{jk}(1) \qquad (4)$$

where j runs over the parity equations for which $h_{jk} \neq 0$, and the scalar α is a normalization factor chosen such that $q_{ik}(0) + q_{ik}(1) = 1$.

Note that an auxiliary buffer is required for storing the values r_{jk} computed in (1) and (2), for the calculation of q_{ik} as defined in equation (3) and (4).

4. Tentative decoding

A single iteration comprises of updating the δq_{ik} based on the outcome of the previous iteration, an application of the horizontal step and then application of the vertical step (except for the first iteration where δq_{ik} is initialized based on the channel measurements). At the end of any iteration (including the initialization step) one can also update the soft-output decision (i.e. the posteriori probability) of each bit:

$$q_k(0) = \alpha\, p_k(0) \prod_j r_{jk}(0) \qquad (5)$$

$$q_k(1) = \alpha\, p_k(1) \prod_j r_{jk}(1) \qquad (6)$$

where j runs over the parity equations for which $h_{jk} \neq 0$. Using (5) and (6), the value of the k`th transmitted bit $\hat{x}_k$ can be estimated as

$$\hat{x}_k = \begin{cases} 1 & q_k(1) > q_k(0) \\ 0 & otherwise \end{cases} \qquad (7)$$

In this manner a tentative bit-by-bit decoding is performed and a vector $\hat{x}$ is obtained. If $H\hat{x} = 0$, i.e. if $\hat{x}$ is a codeword, the decoding algorithm halts declaring $\hat{x}$ as the output. Otherwise, the iterative process continues by returning to Step 2 above. The decoding procedure terminates by declaring a decoding-failure, if some maximum predetermined number of iterations (e.g. 100) occurs with no successful decoding.

As mentioned above, an auxiliary buffer is required for storing the values r_{jk} computed in (1) and (2), for the calculation of q_{ik} as defined in equation (3) and (4). Such a buffer requires a large amount of additional memory and control and thus results in an undesirably high time per iteration. Furthermore, the number of iterations until a result is achieved is also undesirably high. Both of these points render the method unsuitable for use with high volume data such as video and other multimedia related data.

US Patents 6,081,918 and 6,073,250 refer to the use of low density parity check matrices in recovering data that has been lost in transmission. It does not consider the problem of improving decoding efficiency.

US 6,081,909 discusses the use *inter alia* of low density parity check matrices for encoding data. Again it does not deal with the question of efficient decoding.

Summary of the Invention

It is an object of the present invention to overcome the above-mentioned limitations and to provide a method which is suitable for use with high volume data such as multimedia-type data which needs to be processed rapidly. Such a

method may be useful in for example applying multimedia to the forum of wireless communications.

According to a first aspect of the present invention there is a method of decoding LDPC encoded data comprising:

solving parity check equations defined by a low density parity check matrix having a plurality of rows, until parity is reached, for a data matrix iteratively formed by sequential updates of probability differences directly into a single probability difference matrix.

In a preferred embodiment the step of forming a sequential update comprises a step for each row of said parity check matrix of forming a probability difference product

$$S_i = \prod_k \delta q_{ik}$$

where δq_{ik} is the difference between the probabilities that the i^{th} element in the k^{th} row is a "1" and a "0".

A preferred embodiment further comprises the step of forming, for each non-zero entry of the low density parity check matrix:

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} (\delta q_{jk} + S_j) \qquad \text{and}$$

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} (\delta q_{jk} - S_j)$$

Preferably, the probability difference matrix is formed by calculating for each one of a plurality of positions in said matrix the differences between $\boldsymbol{q}(0)$ and $\boldsymbol{q}(1)$ for said respective position.

Preferably there are provided steps of

forming said data matrix {X} using $\{\delta q_{ik}\}$,

testing the data matrix {X} for parity, and

accepting the matrix {X} as a final data matrix if parity is detected.

In an embodiment further iterations are carried out using newly calculated values of δq_{ik} at each stage until parity is detected.

A preferred embodiment comprises the step of ending decoding if parity is not detected after a predetermined number of iterations and declaring that the incoming signal is too corrupted for decoding.

According to a second aspect of the invention there is provided a method of decoding a data signal using a low density parity check matrix, comprising a first stage of iteratively carrying out horizontal and vertical decoding steps to form a matrix $\{\delta q_{ik}\}$ and a second stage of testing for parity, characterized in that the elements of $\{\delta q_{ik}\}$ are updated sequentially, such that each element δq_{ik} is employable for subsequent calculations as soon as it is updated.

The method preferably comprises forming

$$S_i = \prod_k \delta q_{ik}$$

where δq_{ik} is the difference between the probabilities that the i^{th} element in the k^{th} row is a "1" and a "0".

Preferably, the method further comprises the step for each non-zero entry of the low density parity check matrix of forming:

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} (\delta q_{jk} + S_j) \qquad \text{and}$$

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} (\delta q_{jk} - S_j)$$

Preferably, the method further comprises the step of forming a matrix $\{\delta q_{ik}\}$ of differences between $q(0)$ and $q(1)$.

Preferably, the method further comprises the steps of

forming a data matrix {X} using $\{\delta q_{ik}\}$,

testing the data matrix {X} for parity, and

accepting the matrix {X} as a final data matrix if parity is detected.

Preferably the method comprises the steps of iteratively repeating the above steps using the newly calculated values of δq_{ik} at each stage until parity is detected.

Preferably, the step is also provided of ending decoding if parity is not detected after a predetermined number of iterations.

According to a third aspect of the present invention there is provided a decoder for receiving a signal comprising data bits and decoding it using a low density parity check matrix, the decoder comprising:

a probability assignor for assigning to successive incoming data bits probabilities of respectively being "1" and being "0" and for forming a probability difference matrix therefrom,

a probability product difference calculator for producing a probability product difference for each row of said low density parity check matrix,

an updater for updating said respective probability product difference and a respective entry of said probability difference matrix for each non-zero entry of said low density parity check matrix,

a decoder for forming a decoded signal based on said updated probability difference matrix, and

a parity check unit for checking the parity of said decoded signal.

Preferably, the updater is operable to repeat said updating until either one of a group of conditions is fulfilled, said conditions being that said parity check unit indicates that said decoding is successful, and that a predetermined number of repetitions has been reached.

Preferably, the signal is a high density signal.

Preferably, the signal is a multimedia signal.

The signal may be a music signal.

The decoder may be a part of a 3rd generation wireless telephony device.

Brief Description of the Drawings

For a better understanding of the invention, and to show how the same may be carried into effect, reference will now be made, purely by way of example, to the accompanying drawings, in which:

Fig. 1 is a generalized diagram showing a data communication link over a noisy channel,

Fig. 2 is a generalized diagram showing matrices including a sparse matrix, for use in the present invention,

Fig. 3 is a simplified flow diagram showing, as a series of numbered steps, an iterative procedure for decoding using a sparse matrix according to an embodiment of the present invention, and

Fig. 4 is a simplified diagram illustrating the difference between the prior art and embodiments of the invention, in that it eliminates the need to store the matrix "r".

Description of the Preferred Embodiments

According to a first embodiment of the present invention there is provided a procedure for soft-output decoding of low-density parity-check codes. The decoding procedure is based on a principle known as iterative *belief propagation*, as will be described in detail below. In existing algorithms, as described above, two sets of binary distributions are updated iteratively in a process in which all the random variables associated with one set are updated concurrently based on the variables associated with the other set. Thus, an auxiliary buffer is required for the updating mechanism. According to the present embodiment, the updating of the random variables is performed in-place, meaning in the same matrix, thereby avoiding the need for an auxiliary buffer. Moreover, while existing decoding methods employ extensive control mechanisms for administrating the low density matrix, referred to below as the sparse-matrix, the approach of the present embodiments, referred to below as the in-place approach, completely eliminates the need for such control. The in-place approach consequently propagates the updated information much more

rapidly. Thus only a small number of iterations suffice for successful decoding.

Reference is now made to Fig. 1, which is a generalized diagram showing a data communication link over a noisy channel. A sender 10 encodes data using an encoder 12 for sending via a channel 14 to a receiver 16 who decodes the data using a decoder 18. The data signal is distorted as a result of passing through the channel, and noise is added. One of the purposes of encoding is to make the data following modulation more resistant to distortion and to the addition of noise, and to render it susceptible to error detection and correction.

As discussed in detail in the introduction, one of the methods of encoding is the low density parity check matrix, which requires decoding using an iterative process which can become a relatively high consumer of resources if the data being used is particularly dense, tens of thousands of bits in a short period of time, as with video and other multimedia data.

Reference is now made to Fig. 2, which is a generalized diagram showing matrices including a sparse matrix, for use in the present invention. A sparse matrix H represents a series of parity check equations for data X, each equation occupying a row of the matrix, and the application of a parity check based on matrix X should give a result q of zero for error-free data X. The aim of decoding is therefore to solve $H \times X = 0$, for X using an iterative technique.

Reference is now made to Fig. 3, which is a simplified flow diagram showing, as a series of numbered steps, an iterative procedure for decoding

using a sparse matrix according to an embodiment of the present invention. As before, the decoding method is described for the binary case, while the extension of the algorithm to the non-binary case is straightforward.

In the following it is assumed that all codewords are used with equal probability on an arbitrary binary-input continuous-output channel.

1. Initialization 101

Following Gallager's initialization step described above, for each non-zero entry h_{ik} of H set:

$$q_{ik}(0) = p_k(0) \;\; ; \;\; q_{ik}(1) = p_k(1)$$

and we calculate

$$\delta q_{ik} = q_{ik}(0) - q_{ik}(1) .$$

For each parity-check equation i we then calculate

$$S_i = \prod_k \delta q_{ik} \qquad (8)$$

where k runs over all the non-zero entries h_{ik} of the i`th parity-check equation. Thus, S_i is calculated while taking into consideration all the bits that are associated with the i^{th} parity-check equation. The quantity S_i of the i^{th} equation is referred to hereinbelow as the probability difference product of the i^{th} equation.

2. The Horizontal-Vertical Step 102

In this step the algorithm entities are updated and re-employed within the same iteration for the updating process as described below.

For each non-zero entry h_{ik} of H, update q_{ik} in the following manner (step **104**):

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} (\delta q_{jk} + S_j) \tag{9}$$

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} (\delta q_{jk} - S_j) \tag{10}$$

where j runs over the parity equations for which $h_{jk} \neq 0$ excluding the i`th check equation, and where the scalar $\alpha$ is chosen as before. Thus, $q_{ik}$ is updated while taking into consideration all the parity-check equations that include the k`th bit, excluding the i`th equation.

As soon as q_{ik} is calculated in equations (9) and (10), it is then possible to recalculate δq_{ik} and then preferably update S_i as defined in equation (8) using the updated δq_{ik} (step **105**). It is pointed out that the updating of S_i requires merely 2 multiplications regardless of the number of non-zero entries in the i^{th} check equation (and hence it holds true for any LDPC code). Theoretical motivation for the updating method described by equations (9) and (10) is given below, in the section headed "Theory".

3. Tentative decoding

At the end of any iteration (including the initialization step above) the soft-output (i.e. the posteriori probability) for each bit is preferably calculated (step **107**) as follows:

$$q_k(0) = \alpha\, p_k(0) \prod_j (\delta q_{jk} + S_j)$$

$$q_k(1) = \alpha\, p_k(1) \prod_j (\delta q_{jk} - S_j)$$

where j runs over the parity equations for which $h_{jk} \neq 0$. Thus, q_k is calculated while taking into consideration all the parity-check equations that include the k`th bit.

The quantities $\{q_k(1)\}$ and $\{q_k(0)\}$ are preferably used to generate a tentative bit-by-bit decoded vector $\hat{x}$ (step **108**), where each bit $\hat{x}_k$ is given by equation (7). If $H\hat{x} = 0$, i.e. if $\hat{x}$ is a codeword (step **109**), the decoding algorithm terminates successfully (step **110**), declaring $\hat{x}$ as the output. Otherwise, the iterative process continues by returning to the Horizontal-Vertical Step above. The decoding procedure terminates by declaring a decoding-failure (step **112**), if some maximum predetermined number of iterations (e.g. 100) occurs (step **111**) with no successful decoding.

The manner in which the parameters are updated during the horizontal-vertical step **103-106**, is based on the following:

According to the common implementation of belief propagation methods, updating of the probabilities is based on the following formulae:

$$r_{ik}(0) = \frac{1}{2}\left(1 + \prod_{l \neq k} \delta q_{il}\right) \qquad (11)$$

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} r_{jk}(0) \qquad (12)$$

Substituting (11) into (12) yields:

$$q_{ik}(0) = \alpha\, p_k(0) \cdot \prod_{j \neq i} \frac{1}{2}(1 + \prod_{l \neq k} \delta q_{jl})$$

$$= \alpha\, p_k(0) \cdot \prod_{j \neq i} \frac{1}{2 \cdot \delta q_{jk}} \left(\delta q_{jk} + \prod_{l} \delta q_{jl} \right)$$

$$= \alpha\, p_k(0) \cdot \prod_{j \neq i} \frac{1}{2 \cdot \delta q_{jk}} \cdot \prod_{j \neq i} (\delta q_{jk} + \prod_{l} \delta q_{jl}) \qquad (13)$$

In a similar manner:

$$q_{ik}(1) = \alpha\, p_k(1) \cdot \prod_{j \neq i} \frac{1}{2 \cdot \delta q_{jk}} \cdot \prod_{j \neq i} (\delta q_{jk} - \prod_{l} \delta q_{jl}) \qquad (14)$$

Recalling that $S_i = \prod_k \delta q_{ik}$, and since the normalization factor α can absorb the common multiplicands in both (13) and (14), the following equations are obtained

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} (\delta q_{jk} + S_j)$$

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} (\delta q_{jk} - S_j)$$

where α is chosen to satisfy $\alpha q_{ik}(1) + \alpha q_{ik}(0) = 1$.

Hence the simplified updating of the present embodiment becomes possible.

Reference is now made to Fig. 4, which is a simplified diagram illustrating the difference between the prior art and embodiments of the invention, in that it eliminates the need to store the matrix "r". A first process block 200 illustrates the prior art in which the matrix q is converted firstly into a buffer matrix r and the matrix r is then converted into the matrix q. A second process block 202 illustrates the preferred embodiments of the present invention in that the matrix q is converted directly into the matrix q without the need for intermediary buffer matrix r.

In the prior art of LDPC decoding methods, all the relevant parameters, namely $\{\delta q_{ik}\}$, are updated concurrently. Hence, the updating of each element δq_{ik} is based on values that do not change in the course of a single iteration. These values may become stale as more and more elements are updated. By contrast, with in-place decoding, as soon as δq_{ik} is computed, it is used (within the same iteration) for updating all the subsequently computed parameters. This approach propagates the influence of an updated element on the other parameters in a rapid manner. Consequently, the convergence of the iterative algorithm is accelerated. In other words, on the average, less iterations are required for successful decoding.

The above-mentioned result, of accelerated convergence, is difficult to prove analytically, however the empirical results of a simulation, that demonstrate that accelerated convergence in fact occurs, are presented in table

1. The simulation assumes a binary rate ½ LDPC code of block length 13000 bits, and the results are summarized in Table 1. The average number of iterations required for the prior art and the proposed in-place decoding method are compared as a function of the noise level.

E_b/N_0 [dB]	# iterations prior art [Gallager]	# iterations in-place decoding
5.5	4.5	3.1
5.2	5.0	3.2
4.9	5,5	3.7
4.7	6.1	4.0
4.4	7.4	4.3
4.3	10.9	5.4

Table 1: Rate ½ LDPC code, block length 13000 bits, A comparison of the average number of iterations required for convergence

Theory

The following equations show the theoretical justification for the manner in which the parameters are updated during the horizontal-vertical step **103-106** according to the above-described embodiment. According to the common implementation of belief propagation methods, updating of the probabilities is based on the following formulae:

$$r_{lk}(0) = \frac{1}{2}(1 + \prod_{l \neq k} \delta q_{il}) \tag{11}$$

$$q_{lk}(0) = \alpha\, p_k(0) \prod_{j \neq l} r_{jk}(0) \tag{12}$$

Substituting (11) into (12) yields:

$$\begin{aligned} q_{lk}(0) &= \alpha\, p_k(0) \cdot \prod_{j \neq l} \frac{1}{2}(1 + \prod_{l \neq k} \delta q_{il}) \\ &= \alpha\, p_k(0) \cdot \prod_{j \neq l} \frac{1}{2 \cdot \delta q_{jk}} \left(\delta q_{jk} + \prod_{l} \delta q_{jl} \right) \end{aligned}$$

$$= \alpha\, p_k(0) \cdot \prod_{j \neq l} \frac{1}{2 \cdot \delta q_{jk}} \cdot \prod_{j \neq l} (\delta q_{jk} + \prod_{l} \delta q_{jl}) \tag{13}$$

In a similar manner:

$$q_{lk}(1) = \alpha\, p_k(1) \cdot \prod_{j \neq l} \frac{1}{2 \cdot \delta q_{jk}} \cdot \prod_{j \neq l} (\delta q_{jk} - \prod_{l} \delta q_{jl}) \tag{14}$$

Recalling that $S_l = \prod_k \delta q_{lk}$, and since the normalization factor α can absorb the common multiplicands in both (13) and (14), the following equations are obtained

$$q_{lk}(0) = \alpha\, p_k(0) \prod_{j \neq l} (\delta q_{jk} + S_j)$$

$$q_{lk}(1) = \alpha\, p_k(1) \prod_{j \neq l} (\delta q_{jk} - S_j)$$

where α is chosen to satisfy $\alpha q_{ik}(1)+\alpha q_{ik}(0)=1$. Thus updating of the matrix according to the above-described procedure becomes possible.

The above embodiments thus improve upon prior art iterative decoding methods for LDPC codes. The main procedural difference is in the process of a single iteration as described above. An array $\{\delta q_{ik}\}$ is used to pass information from one iteration to the next. As noted in Step 3 of Gallager's algorithm, the elements in this array need to be updated in each iteration. In the prior art, all the elements are updated concurrently based on equations (3) and (4). This requires an auxiliary buffer for storing the values $\{r_{ik}\}$ computed in equations (1) and (2). By contrast, according to the above-described embodiments, the elements of $\{\delta q_{ik}\}$ are updated sequentially and each element $\{\delta q_{ik}\}$ is employed for subsequent calculations as soon as it is updated. The methods of the preferred embodiments are hence referred to as *in-place* decoding. The preferred embodiments have several inherent advantages:

1) A small buffer, whose size is proportional to the number of parity-check equations, is employed, rather than the one required for saving the data $\{r_{ik}\}$ whose size is proportional to the number of non-zero elements in H.

2) The excessive control mechanism required for administrating a large sparse matrix is preferably avoided due to the sequential nature of the in-place procedure.

3) Updated information preferably propagates rapidly through the graph representation of the code and therefore the computed probabilities are more accurate.

4) The average number of iterations required for successful decoding and hence the decoding delay and the overall decoding complexity (operations per decoded bit) is preferably considerably reduced.

There is thus provided a method of decoding of received signals which have been encoded using LDPC matrices, which method does away with an intermediate matrix and which leads to earlier convergence of decoding iterations.

It is appreciated that features described only in respect of one or some of the embodiments are applicable to other embodiments and that for reasons of space it is not possible to detail all possible combinations. Nevertheless, the scope of the above description extends to all reasonable combinations of the above described features.

The present invention is not limited by the above-described embodiments, which are given by way of example only. Rather the invention is defined by the appended claims.

Claims

We claim:

1. A method of decoding LDPC encoded data comprising

solving parity check equations defined by a low density parity check matrix having a plurality of rows, until parity is reached, for a data matrix iteratively formed by sequential updates of probability differences directly into a single probability difference matrix.

2. A method according to claim 1, wherein forming a sequential update comprises a step for each row of said parity check matrix of forming a probability difference product $S_i = \prod_k \delta q_{ik}$

where δq_{ik} is the difference between the probabilities that the i^{th} element in the k^{th} row is a "1" and a "0".

3. A method according to claim 2, further comprising the step for each non-zero entry of the low density parity check matrix of forming:

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} (\delta q_{jk} + S_j)$$ and

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} (\delta q_{jk} - S_j)$$

4. A method according to claim 3, wherein said probability difference matrix is formed by calculating for each one of a plurality of positions in said matrix the differences between $q(0)$ and $q(1)$ for said respective position.

5. A method according to claim 3, further comprising the steps of

forming said data matrix {X} using $\{\delta q_{ik}\}$,

testing the data matrix {X} for parity, and

accepting the matrix {X} as a final data matrix if parity is detected.

6. A method according to claim 5, comprising the steps of repeating the steps of claims 2 to 5 using the newly calculated values of δq_{ik} if parity is not detected.

7. A method according to claim 6, further comprising the step of ending decoding if parity is not detected after a predetermined number of iterations.

8. A method of decoding a data signal using a low density parity check matrix, comprising a first stage of iteratively carrying out horizontal and vertical decoding steps to form a matrix $\{\delta q_{ik}\}$ and a second stage of testing for parity, characterized in that the elements of $\{\delta q_{ik}\}$ are updated sequentially,

such that each element δq_{ik} is employable for subsequent calculations as soon as it is updated.

9. A method according to claim 8, comprising forming

$$S_i = \prod_k \delta q_{ik}$$

where δq_{ik} is the difference between the probabilities that the i^{th} element in the k^{th} row is a "1" and a "0".

10. A method according to claim 9, further comprising the step for each non-zero entry of the low density parity check matrix of forming:

$$q_{ik}(0) = \alpha\, p_k(0) \prod_{j \neq i} (\delta q_{jk} + S_j) \qquad \text{and}$$

$$q_{ik}(1) = \alpha\, p_k(1) \prod_{j \neq i} (\delta q_{jk} - S_j)$$

11. A method according to claim 10, further comprising the step of forming a matrix $\{\delta q_{ik}\}$ of differences between $\boldsymbol{q}(0)$ and $\boldsymbol{q}(1)$.

12. A method according to claim 11, further comprising the steps of

forming a data matrix {X} using $\{\delta q_{ik}\}$,

testing the data matrix {X} for parity, and

accepting the matrix {X} as a final data matrix if parity is detected.

13. A method according to claim 12, comprising the steps of repeating the steps of claims 2 to 5 using the newly calculated values of δq_{ik} if parity is not detected.

14. A method according to claim 13, further comprising the step of ending decoding if parity is not detected after a predetermined number of iterations.

15. A decoder for receiving a signal comprising data bits and decoding it using a low density parity check matrix, the decoder comprising:

a probability assignor for assigning to successive incoming data bits probabilities of respectively being "1" and being "0" and for forming a probability difference matrix therefrom,

a probability product difference calculator for producing a probability product difference for each row of said low density parity check matrix,

an updater for updating said respective probability product difference and a respective entry of said probability difference matrix for each non-zero entry of said low density parity check matrix,

a decoder for forming a decoded signal based on said updated probability difference matrix, and

a parity check unit for checking the parity of said decoded signal.

16. A decoder according to claim 15, wherein said updater is operable to repeat said updating until either one of a group of conditions is fulfilled, said conditions being that said parity check unit indicates that said decoding is successful, and that a predetermined number of repetitions has been reached.

17. A decoder according to claim 15, wherein said signal is a video signal.

18. A decoder according to claim 15, wherein said signal is a multimedia signal.

19. A decoder according to claim 15, wherein said signal is a music signal.

20. A decoder according to claim 15, which is part of a 3rd generation wireless telephony device.

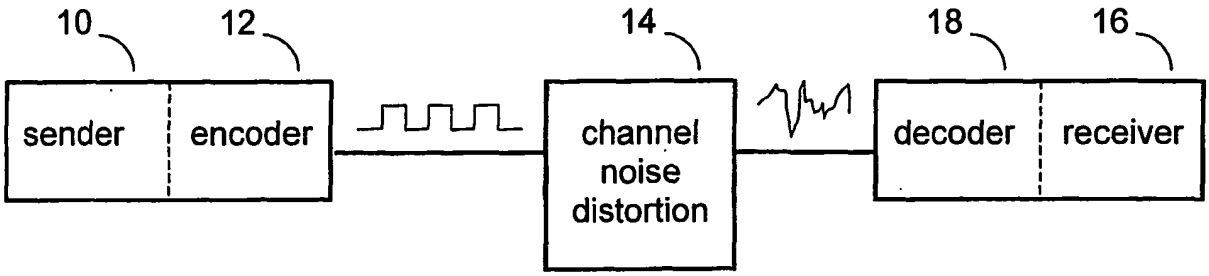
10
12
14
18
16
sender
encoder
channel
noise
distortion
decoder
receiver

Fig. 1

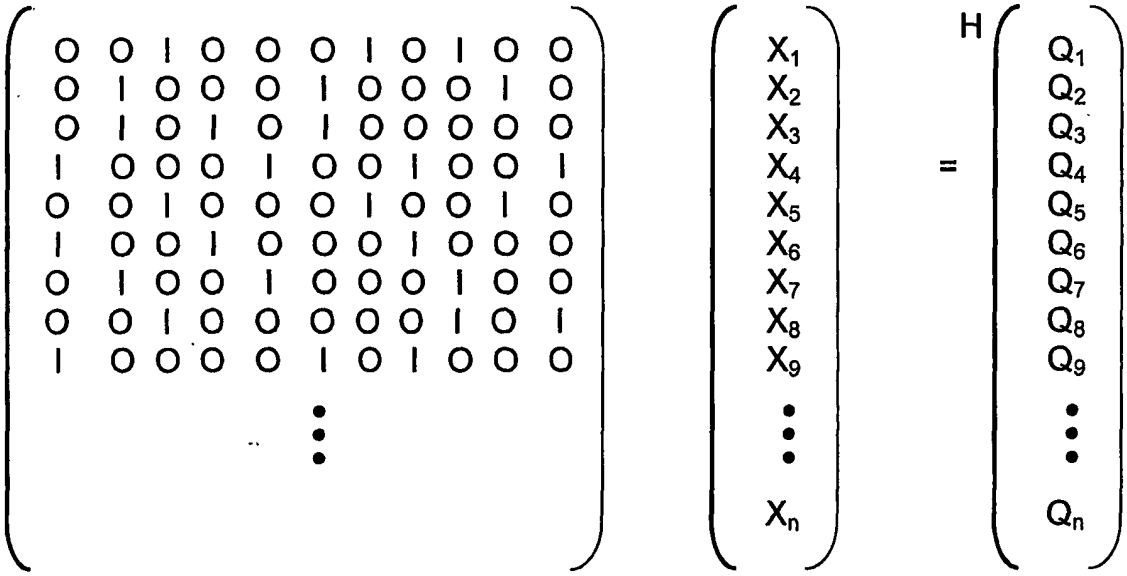
H
=

Fig. 2

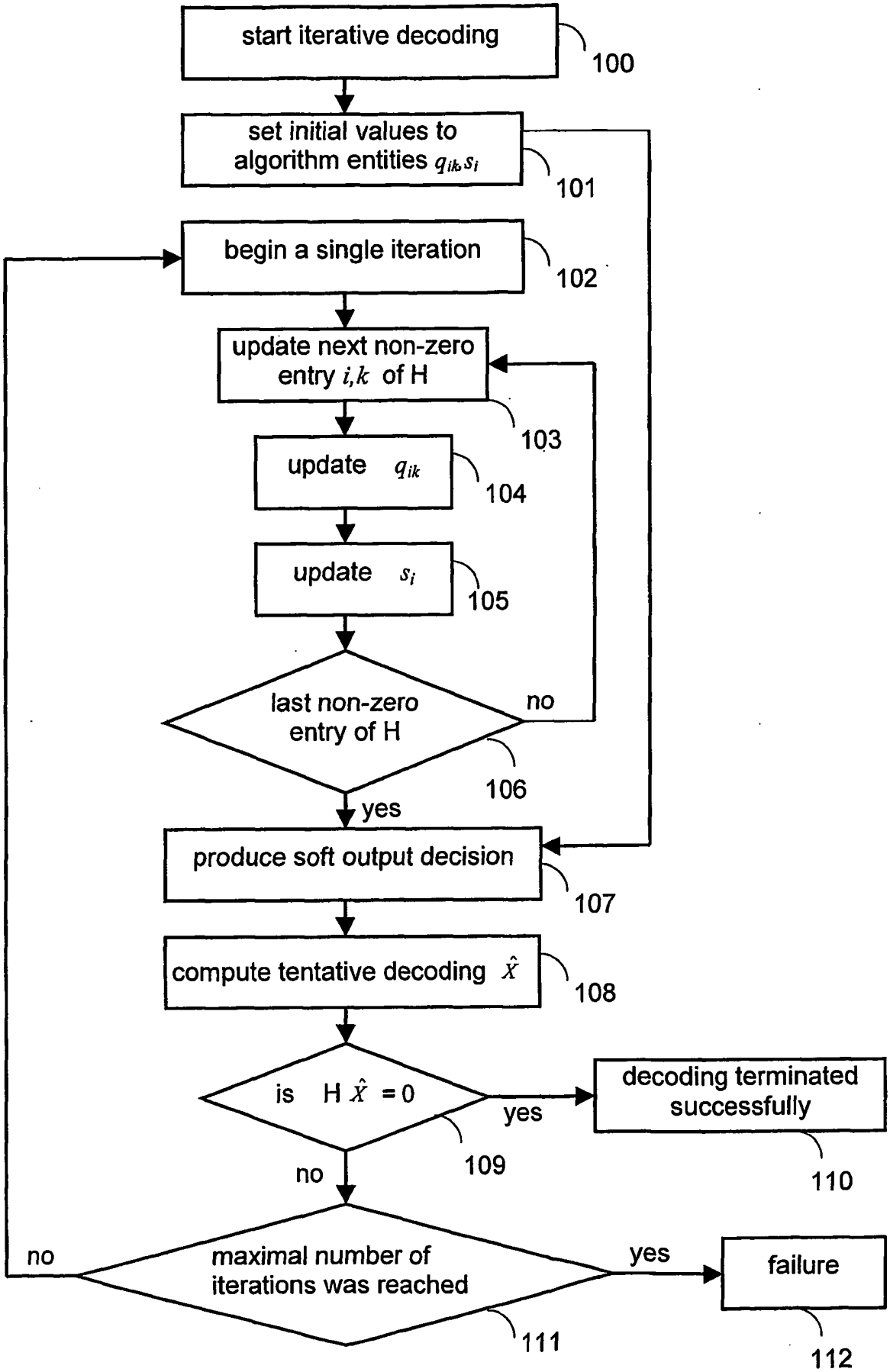
start iterative decoding
100
set initial values to algorithm entities q_{ik}, s_i
101
begin a single iteration
102
update next non-zero entry i,k of H
103
update q_{ik}
104
update s_i
105
last non-zero entry of H
no
106
yes
produce soft output decision
107
compute tentative decoding $\hat{X}$
108
is $H\hat{X} = 0$
yes
decoding terminated successfully
110
109
no
no
maximal number of iterations was reached
yes
failure
111
112

Fig. 3

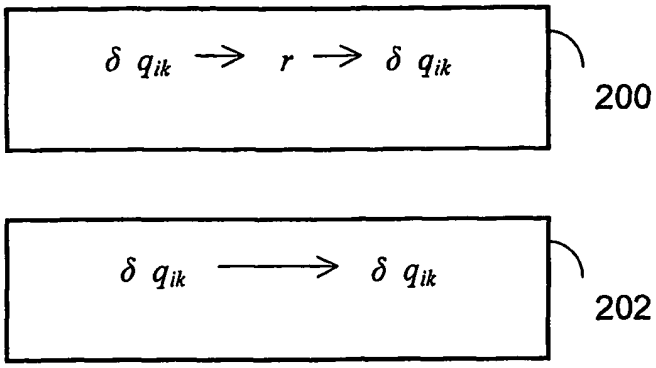
$\delta\ q_{ik} \rightarrow r \rightarrow \delta\ q_{ik}$
200
$\delta\ q_{ik} \longrightarrow \delta\ q_{ik}$
202

Fig. 4